Security Flaws in IoT Devices: Investigation and Defense Mechanisms

Olumide Kayode
Department of Computer Science, University of Texas at San Antonio
One UTSA Circle, San Antonio, TX 78249, United States

Abstract

The proliferation of Internet of Things (IoT) devices has led to massive generation of sensor data and increase in attack vectors. Traffic generated by these devices require more exploratory studies in order to determine the effective ways to secure transmitted data and avert exploitation by hackers. IoT devices susceptibility to direct or remote manipulation, secure channel used for message transmission and resilience of the underlying middleware (broker) to exploitation are very important issues that worth consideration. With more data ever being collected than before, the security of user's information and devices identities are becoming a great concern. In this research work, we evaluate the vulnerabilities inherent in an IoT ecosystem design that was developed using Raspberry Pi, sensors and MQTT protocol. We also developed a real-time anomaly detection in sensor reading to avert dangerous situation and evaluate the effectiveness of our work.

Keywords: Internet of Things, Vulnerabilities, Data privacy, Security mechanisms
DOI: 10.7176/JIEA/10-1-04
Publication date: January 31st 2020

1. Introduction

The current era of Internet of Things (IoT) has led to the development of many IoT devices and their adoption into various aspect of industries, institutions and the society at large. The emergence of smart cars, smart fridge, smart locks, smart plug to mention a few are evidence of the proliferation of IoT. Sensors will arguably play a major role in the IoT ecosystem for interacting with the physical entities and transmission of data, both digital and analog sensor data. With the massive sensor data that are being generated, the security and reliability of IoT ecosystem is of utmost importance. Users are not fully convinced and still skeptical about the security of their information. Many recent cyber attacks and exploitation of vulnerabilities in connected devices have heighten user's concerns. Inherent vulnerabilities in devices like outdated firmware, unencrypted data transmission and insecure communication channels can easily be exploited to compromise the security of IoT devices. It has been discovered that many manufacturers of IoT devices do not often consider robust security mechanism during the design phase and implementation process. The size and resource-constrained nature of IoT devices are some of the major factors that has been identified which limit the integration of such security mechanism. Lightweight communication protocols and cryptographic algorithms have been suggested as a viable solution to improve the security of IoT devices.

Moreover, ubiquitous data generated by IoT devices have increased the means and attack vectors that enable hackers to exploit device vulnerabilities. Security flaws in network access points, routers and gateways existing between the communicating IoT devices, smart phones used for user's interaction and target servers can be exploited. Data transmitted in plain text can be intercepted while encrypted data can also be decrypted using suitable algorithm to decipher coded data and reveal sensitive information. Data security and privacy of IoT remains a major challenge and will be the main focus in the foreseeable future. A survey on IoT security and device vulnerability discussed various security issues like intrusion detection, threat modeling and ever-evolving IoT vulnerabilities (Neshenko *et al,* 2019). The authors identified numerous security breaches, attacks which exploit vulnerabilities and corresponding remediation methodologies. One of such recent attacks is the Marai botnet which affected major internet platforms and cloud services. The distributed denial-of-service attack primarily target devices like IP cameras and home routers. Devices connected to such compromised routing devices can easily be exposed to malicious software or malware which will further propagate the devasting effect of the implemented attack.

In this research work, we developed an IoT system using Raspberry Pi, sensors and Message Queuing Telemetry Transport (MQTT) protocol for publish-subscribe communication. We utilized three Raspberry Pi devices that communicate and transmit data via MQTT during interaction with the sensors. We further discuss active and passive attacks which can be used to compromise the security of the IoT system. Furthermore, we also discuss ways through which any abnormality in the normal behavior or trends of sensors data can be detected.

2. Related Work

Several works have studied the security issues in IoT and discussed possible solutions. One of such works (Hameed & Alomary 2019) was presented in a survey which reviewed relevant algorithms that can be used to secure IoT networks. The authors discussed authentication methods and hardware security support for IoT. A broad overview of the security risks in IoT and possible counteractions was also discussed in a recent survey (Meneghello *et al* 2019). Specific security mechanisms adopted by most popular IoT communication protocols was presented and the authors analyzed some of the attacks against IoT devices. Safety necessities like secrecy, unity and substantiation for IoT was presented in another research work (Gurunath *et al* 2018). IoT threat analysis involving twelve kind of different attacks, use of botnet, security issues in IoT networks and power consumption issues were also discussed.

An IoT security model that can be used by organization to plan a strategy and discourse about IoT security from end-to-end perspective was presented in a recent conference on pervasive computing and communications (Bugeja *et al* 2019). The authors proposed a conceptual framework that can be used to analyze, describe and measure overall security level of an IoT ecosystem. Concepts like security-by-design processes and standards, continuous and automated risk assessment, data and application threat modeling, security testing, continuous monitoring, and auditing were discussed in relation to an organization’s security practices. A methodology for the development of a consumer security index (CSI) to aid consumer decision making and encourage greater security provision in the manufacture of IoT devices was suggested in a research work (Blythe *et al* 2018). The authors employed an online survey to identify consumer preferences and identified security features that consumer IoT devices should provide. They also developed a matrix of different classes of IoT devices and explored the use of natural language processing to extract data from device user manuals to identify security features provided by manufacturers. In one of our earlier works (Olumide & Tosun 2019), we explored the data being transmitted by six representative IoT devices and analyzed the data using a proxy server to capture both Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) traffic. Sensitive information like user’s information and IP addresses were identified in our data analysis. We suggested that IoT devices should not allow proxy connections and developed a machine learning algorithm to detect proxies using network connection information.

The need for a systematic and automated methodology that enables scalable testing approaches for security aspects in IoT ecosystem was discussed in a research paper on a project work that aimed to give scientists the ability to assess and compare different IoT security technologies (García *et al* 2018). The authors proposed a test-based risk assessment and security certification for IoT. A middleware architecture which provides an end-to-end security solution for contributors who upload sensing data was suggested in another recent work (Garg & Dave 2019). The proposed approach allows an end to end encryption of data to secure information in transit. The authors utilize a Representational State Transfer (REST) application programming interface (API) to communicate and exchange data. This provided an interface for user to register their IoT devices and securely access data collected by the device. During the process of collecting IoT information, the type and timing of the shared information so as to ensure confidentiality of the shared information and establish the rating of information sharing has also been addressed (Choi *et al* 2018). The authors discussed a functional requirement in the IoT information security sharing system and established functions to be performed between individuals in the reference model of the IoT information security sharing system. Security concerns like device cloning, sensitive data exposure, data tampering, denial of service, unauthorized device access and control were also discussed in (Naik & Maral 2017). However, the authors only focused on ways to mitigate IoT security challenges pertaining to device cloning and sensitive data exposure.

3. Security Vulnerabilities in IoT

3.1. IoT System Design

In this work, we first developed an IoT system that monitors weather attributes so that users can decide if any extra time is needed for their commute. The system uses real-time processing of a combination of imagery and sensor data to fully describe the weather condition to the users. The users can then access all of this data via a MQTT app on their smartphone.

The weather sensor client Pi’s purpose is to collect real-time weather data and transmit via MQTT to the broker Pi for further processing. The five different sensors on this Pi are a digital temperature sensor, digital humidity sensor, analog smoke detector, analog combustible gas detector, and analog CO detector. The digital sensors connect directly to the General-Purpose Input/Output (GPIO) pins of the Raspberry Pi while the analog sensors connect to an Analog to Digital (A2D) converter which then connects to the Raspberry Pi. The A2D converter is required since all of the GPIO pins on the Raspberry Pi are digital pins. The Pi will read the sensors and determine if the thresholds for the analog sensors has been exceeded. The Pi will then configure this information into a predetermined JavaScript Object Notation (JSON) format and transmit this JSON over MQTT to the broker Pi with the topic “iot/sensor_data” for further processing. This whole process will happen every 10 seconds.

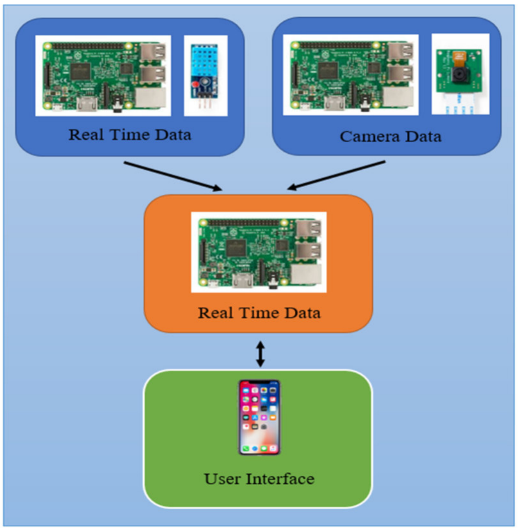
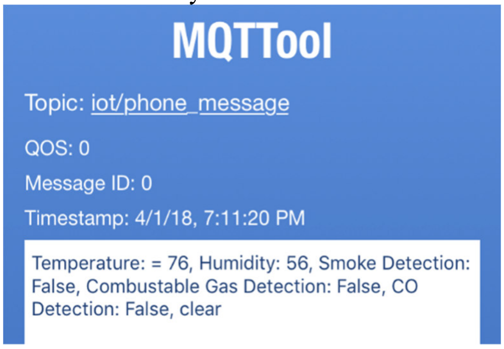


Fig. 1: IoT system architecture

The camera client Pi as shown in figure 1 serves the purpose of image capturing and its analysis. It is used for visual confirmation of the atmospheric disposition of the area where the sensors are utilized for monitoring purposes. Using our python script, we automate image capturing via the Pi camera every 10 seconds and use the Google cloud platform for the image analysis. The cloud assisted method via the Google cloud platform helps in object identification in captured images using machine learning techniques. This helps in our evaluation of the current disposition of the atmospheric condition visually.



MQTTool

Topic: iot/phone_message

QOS: 0

Message ID: 0

Timestamp: 4/1/18, 7:11:20 PM

Temperature: = 76, Humidity: 56, Smoke Detection: False, Combustable Gas Detection: False, CO Detection: False, clear

Fig 2: Phone MQTT message

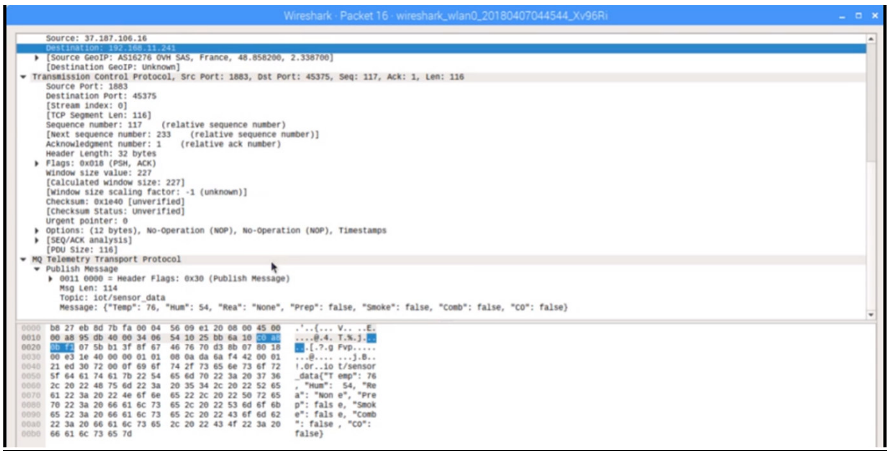
The broker Pi's purpose is to collect the data that is being published from the weather sensor Pi with the topic "iot/sensor_data" and the camera sensor Pi with the topic "iot/camera_data" and make it where the user can gain access to this data via MQTT. When the user sends the word "data" to the topic "iot/get_command" the broker Pi will then send the data back to the phone via MQTT with the topic "iot/phone_message." as shown in figure 2.

3.2. Cyber Flaws Identification and Exploitation

Here, the purpose is to discover and then exploit vulnerabilities that will allow an adversary to alter any data that the system collects and sends. Also, a machine learning technique will be utilized to identify if any of the sensor data has been tampered with or modified.

Eavesdropping on the Broker Pi

Once an adversary has access to the network, they could use Wireshark to start collecting information about any devices on the network, especially the IoT devices. Here, Wireshark was installed directly on the broker IoT device itself so the client sensor data, client camera data, and user interaction with the client could be monitored.



Wireshark · Packet 16 · wireshark_wlan0_20180407044544_Xv96Ri

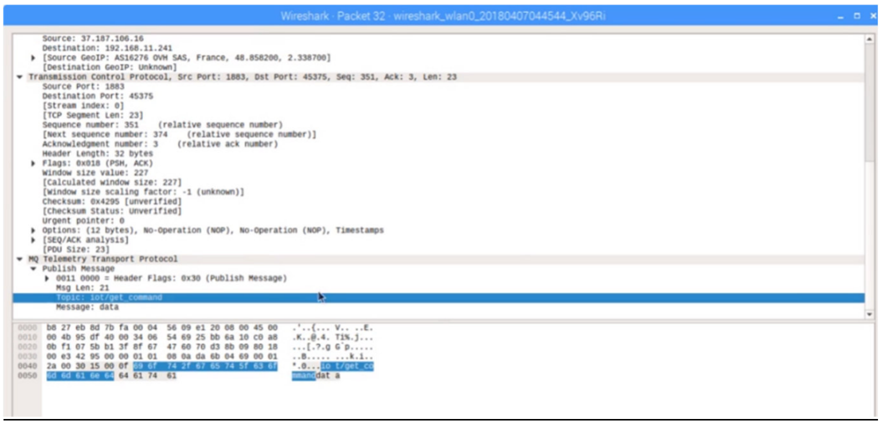
```
      Source: 37.187.106.16
      Destination: 192.168.11.241
    ▸ [Source GeoIP: AS16276 OVH SAS, France, 48.858200, 2.338700]
      [Destination GeoIP: Unknown]
▾ Transmission Control Protocol, Src Port: 1883, Dst Port: 45375, Seq: 117, Ack: 1, Len: 116
      Source Port: 1883
      Destination Port: 45375
      [Stream index: 0]
      [TCP Segment Len: 116]
      Sequence number: 117    (relative sequence number)
      [Next sequence number: 233    (relative sequence number)]
      Acknowledgment number: 1    (relative ack number)
      Header Length: 32 bytes
    ▸ Flags: 0x018 (PSH, ACK)
      Window size value: 227
      [Calculated window size: 227]
      [Window size scaling factor: -1 (unknown)]
      Checksum: 0x1e40 [unverified]
      [Checksum Status: Unverified]
      Urgent pointer: 0
    ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▸ [SEQ/ACK analysis]
      [PDU Size: 116]
▾ MQ Telemetry Transport Protocol
    ▾ Publish Message
        ▸ 0011 0000 = Header Flags: 0x30 (Publish Message)
          Msg Len: 114
          Topic: iot/sensor_data
          Message: {"Temp": 76, "Hum": 54, "Rea": "None", "Prep": false, "Smoke": false, "Comb": false, "CO": false}

0000  b8 27 eb 8d 7b fa 00 04  56 09 e1 20 08 00 45 00   .'..{... V.. ..E.
0010  00 a8 95 db 40 00 34 06  54 10 25 bb 6a 10 c0 a8   ....@.4. T.%.j...
0020  0b f1 07 5b b1 3f 8f 67  46 76 70 d3 8b 07 80 18   ...[.?.g Fvp.....
0030  00 e3 1e 40 00 00 01 01  08 0a da 6a f4 42 00 01   ...@.... ...j.B..
0040  21 ed 30 72 00 0f 69 6f  74 2f 73 65 6e 73 6f 72   !.0r..io t/sensor
0050  5f 64 61 74 61 7b 22 54  65 6d 70 22 3a 20 37 36   _data{"T emp": 76
0060  2c 20 22 48 75 6d 22 3a  20 35 34 2c 20 22 52 65   , "Hum":  54, "Re
0070  61 22 3a 20 22 4e 6f 6e  65 22 2c 20 22 50 72 65   a": "Non e", "Pre
0080  70 22 3a 20 66 61 6c 73  65 2c 20 22 53 6d 6f 6b   p": fals e, "Smok
0090  65 22 3a 20 66 61 6c 73  65 2c 20 22 43 6f 6d 62   e": fals e, "Comb
00a0  22 3a 20 66 61 6c 73 65  2c 20 22 43 4f 22 3a 20   ": false , "CO":
00b0  66 61 6c 73 65 7d                                  false}
```

Fig 3: Sensor client MQTT transmission

A filter was applied to the incoming data so only the MQTT messages could be observed as shown in figure 3. The MQTT messages that were observed include the broker Pi ping request to the online MQTT server, the data being sent from the camera client to the gateway Pi, the sensor data being sent from the sensor client Pi to the gateway Pi, and the user request from the MQTT client to the gateway Pi. There is so much useful information that an adversary can collect from just this one MQTT message. The available information that will be exposed include IP address of client Pi, IP address of broker Pi, port number MQTT message is sent through, MQTT topic and client Pi data.

The figure 4 shows the data gathered about the user's request acquired through eavesdropping. The advantage of using an online broker instead of installing the broker on the Raspberry Pi itself is that the user's IP address is not given in the transaction, thus protecting the privacy of the user. All of the information collected will later be used to find other vulnerabilities and launch attacks.



Wireshark · Packet 32 · wireshark_wlan0_20180407044544_Xv96Ri

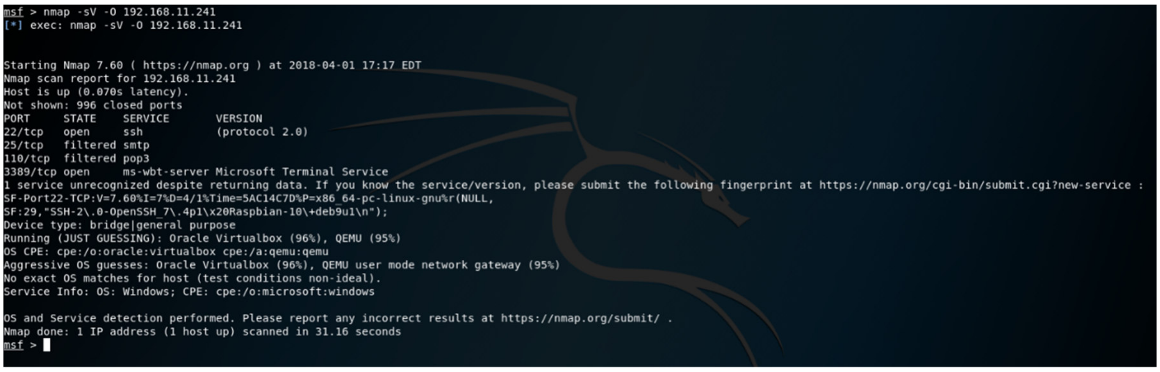
```
      Source: 37.187.106.16
      Destination: 192.168.11.241
    ▸ [Source GeoIP: AS16276 OVH SAS, France, 48.858200, 2.338700]
      [Destination GeoIP: Unknown]
▾ Transmission Control Protocol, Src Port: 1883, Dst Port: 45375, Seq: 351, Ack: 3, Len: 23
      Source Port: 1883
      Destination Port: 45375
      [Stream index: 0]
      [TCP Segment Len: 23]
      Sequence number: 351    (relative sequence number)
      [Next sequence number: 374    (relative sequence number)]
      Acknowledgment number: 3    (relative ack number)
      Header Length: 32 bytes
    ▸ Flags: 0x018 (PSH, ACK)
      Window size value: 227
      [Calculated window size: 227]
      [Window size scaling factor: -1 (unknown)]
      Checksum: 0x4295 [unverified]
      [Checksum Status: Unverified]
      Urgent pointer: 0
    ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▸ [SEQ/ACK analysis]
      [PDU Size: 23]
▾ MQ Telemetry Transport Protocol
    ▾ Publish Message
        ▸ 0011 0000 = Header Flags: 0x30 (Publish Message)
          Msg Len: 21
          Topic: iot/get_command
          Message: data

0000  b8 27 eb 8d 7b fa 00 04  56 09 e1 20 08 00 45 00   .'..{... V.. ..E.
0010  00 4b 95 df 40 00 34 06  54 69 25 bb 6a 10 c0 a8   .K..@.4. Ti%.j...
0020  0b f1 07 5b b1 3f 8f 67  47 60 70 d3 8b 09 80 18   ...[.?.g G`p.....
0030  00 e3 42 95 00 00 01 01  08 0a da 6b 04 69 00 01   ..B..... ...k.i..
0040  2a 00 30 15 00 0f 69 6f  74 2f 67 65 74 5f 63 6f   *.0...io t/get_co
0050  6d 6d 61 6e 64 64 61 74  61                        mmanddat a
```

Fig 4: User client transmission information

Vulnerability 1: Default Password/Configuration of Raspberry Pi

First, Network Mapper (Nmap) scan of the of the IP address was implemented to see the version of the network processors and thus determine what vulnerabilities are in existence. Once the data has been received, a Nessus vulnerability scan was implemented to see what kind of exploits can be used on the system. Specifically, the vulnerability that was searched for entails using the default username and password to access the system when the Secure Shell (SSH) port is open. Using a tool named Metasploit on the Kali Linux operating system, Nmap scan was executed on both the client and gateway Pi to search for vulnerabilities. Using the command: *nmap -sV -O 192.168.11.212*, on the client Pi and *nmap -sV -O 192.168.11.241* on the



```
msf > nmap -sV -O 192.168.11.241
[*] exec: nmap -sV -O 192.168.11.241


Starting Nmap 7.60 ( https://nmap.org ) at 2018-04-01 17:17 EDT
Nmap scan report for 192.168.11.241
Host is up (0.070s latency).
Not shown: 996 closed ports
PORT     STATE    SERVICE       VERSION
22/tcp   open     ssh           (protocol 2.0)
25/tcp   filtered smtp
110/tcp  filtered pop3
3389/tcp open     ms-wbt-server Microsoft Terminal Service
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port22-TCP:V=7.60%I=7%D=4/1%Time=5AC14C7D%P=x86_64-pc-linux-gnu%r(NULL,
SF:29,"SSH-2\.0-OpenSSH_7\.4p1\x20Raspbian-10\+deb9u1\n");
Device type: bridge|general purpose
Running (JUST GUESSING): Oracle Virtualbox (96%), QEMU (95%)
OS CPE: cpe:/o:oracle:virtualbox cpe:/a:qemu:qemu
Aggressive OS guesses: Oracle Virtualbox (96%), QEMU user mode network gateway (95%)
No exact OS matches for host (test conditions non-ideal).
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.16 seconds
msf > 
```

Fig 5: Nmap Scan of Broker Pi 192.168.11.241

gateway Pi. The port was not open on the client Pi but it was open on the broker Pi as shown in figure 5. The first vulnerability that was discovered in the system was a default username and password that is used for the SSH login from a computer to a Raspberry Pi was still active. To exploit this vulnerability, the adversary has to be in the same internal network as the system that was intended to be exploited. If the adversary was on an external network, the remote SSH login could not be achieved. The issue with this vulnerability is that once the hackers connects with the SSH login, they have full access to the system and have the potential to alter any data, software, or setting of the IoT devices. In our experiment, using the default username: pi and default password: raspberry, we gained access to the system.

Vulnerability 2: Unencrypted MQTT Messages

Another vulnerability of the system that was discovered is the transmission of unencrypted MQTT messages between the IoT devices. An adversary can easily interpret the MQTT message payload. One attack that was done was to create a compromised IOT sensor client to send incorrect data to the gateway. This would be easy for the adversary to create since they could view the data structure that the sensory information is send and recreate it with incorrect information. To exploit this vulnerability, MQTT messages were sent using another Raspberry Pi with incorrect temperature values, such as the temperature being 5 degrees instead of 74 degrees, in the required JSON format to the gateway Pi with the topic “iot/sensor_data” will give the user inaccurate data. This type of attack represented an adversary gaining access to the network and creating adding a compromised client into the system with the purpose to give the user incorrect data.

In addition to the false data being sent, a DOS attack could be launched to not allow the correct data to be introduced to the system making it where only the incorrect data will be viewed. To replicate this type of attack, the additional Raspberry Pi was programmed to send the incorrect data in the required JSON format continuously to the gateway Pi with the topic “iot/sensor_data”. This accomplished the task where the actual sensor data was never received by the gateway Pi. Even if it was received, it would be replaced by the incorrect data instantaneously.

Vulnerability 3: Physical Attack

Assuming that the adversary has access to the system they could launch a physical attack on the system. As shown in figure 6, all of the electrical components on the breadboard are connected to a 3.3 Volt Power source.

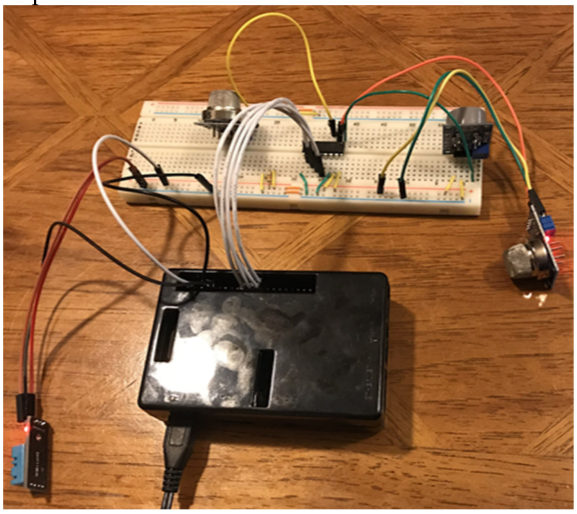
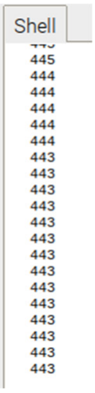
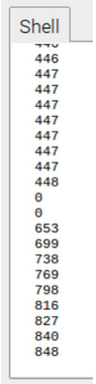


Fig 6: Vulnerable Physical System

One method a physical attack could be implemented is that the analog sensors could be disconnected from the 3.3 Volt source and connect them to the 5 Volt source on the Raspberry Pi while still leaving the A2D converter connected to the 3.3 Volt source. The output data from the sensors as shown in figure 7, will give higher reading than the average, making the sensor reading inaccurate.



```
445
444
444
444
444
444
443
443
443
443
443
443
443
443
443
443
443
443
443
443
```



```
446
447
447
447
447
447
447
447
448
0
0
653
699
738
769
798
816
827
840
848
```

Fig 7a: Correct 3.3 Volt CO_2 Sensor Measurement **Fig 7b:** 5.5 Volt Inflated CO_2 Sensor Measurement

Another way a physical attack could be implemented involves wiring the analog input of the A2D converter directly to the 3.3 Volt power supply. This will result in a situation where the sensor would always give a false detection of CO_2 gas. Arguably, the most dangerous method a physical attack could be implemented entails connecting the analog input of the A2D converter directly to ground. This will make it unable to detect CO_2 gas which could possibly harm user. All of these physical attacks would affect the reliability of the system.

3.3. System Protection Measures

As earlier identified, there are multiple vulnerabilities and attacks that can be implemented on the system. However, majority of these attacks can be prevented by adding protection measures and security mechanisms. *Default Password*

This is actually the simplest solution to fix one of the vulnerabilities. This involves creating a new user and a unique password for that user. If the adversary wanted to gain access to the system, they will have to find the password through brute force or other password cracking techniques. Once this is done, an adversary cannot use the default username and password to access the system.

SSH Login

Another option to eliminate vulnerability is to turn off SSH login on the Pi or close the port all together. The SSH login ability was turned off once our experiments was completed so the adversary could not remote login to the system.

Hardware Interference

The hardware interference vulnerability can be solved by fabricating the circuit on a circuit board and enclosing the circuit in a case. This will increase the difficulty of creating hardware interference instead of just rewiring the circuit on the breadboard. A software prevention method would be to add a sensor which monitor the quality of transmitted data. The system will send an alert when the sensor data output is below a certain threshold, such as when the output of the sensor is grounded. The same will happen when the sensor data output is above the average by a certain threshold for a certain amount of time. For example, when the output of the sensor was wired high or 5 volts is applied to the sensor instead of 3.3 Volt thus inflating the values.

MQTT Interference

One prominent way to secure MQTT messages that are transmitted in plain text is to utilize encryption. Once the data was converted into JSON block format on the sensor client Pi, the messages would be encrypted and then sent over MQTT to the gateway Pi. The gateway Pi will then decrypt the messages and store the sensor data. A simple and least computationally expensive encryption algorithm involves converting the ASCII text in the string to hexadecimal (hex). A keyword string is used to encrypt the messages. The number of characters is counted in that keyword string and this number is subtracted from every hex character in MQTT message, thus creating an encrypted message. When a message is received by the broker Pi, the characters are counted in the keyword string and the number is added to every hex character in the MQTT message thus decrypting the message. This approach accomplished the goal of making it difficult for adversary to view data in plain text and easily craft an attack.

4. Real-time Anomaly Detection

Towards an intelligent security mechanism, an intrusion detection method using neural network to determine whether the sensor's state or data has been tampered with or altered was implemented. The features in the sensor data are Temperature, Humidity, Smoke detection, Combustible gas detection, CO detection, and Picture. Only

Temperature and Humidity, out of the six features have numerical values. Hence, the need to encode the non-numeric values for the other four features into numeric values for machine learning computation. Since the values return by Smoke detection, Combustible gas detection and CO detection can either be “True” or “False”, we encode 1 for “True” and -1 for “False”. Concerning Picture, it can either return “Clear” or “Smoky”. We encode -1 for “Clear” and 1 for “Smoky”. A typical vector representation of the sensor reading looks like [76, 56, -1, -1, -1, -1] which are [Temperature, Humidity, Smoke detection, Combustible gas detection, CO detection, and Picture]. Concerning the huge numeric values for Temperature and Humidity which may outwit the other encoded values, we need to scale down the values so as to be in a range close to the encoded values. We performed feature scaling to reduce the huge numbers to values within the range 0 and 1, using the formula x = (x - xmin) / (xmax - xmin). The output based on the sensor’s data can either be “Good” encoded with 0 or “Danger” encoded with 1. Using various possible combination, we formulate the training data. For example, a sample of the training data will look like [0.72, 0.58, -1, -1, -1, -1, 0], where the first six values will be for the xtrain data and last value i.e. 0 will be for the ytrain. We utilized the Multilayer Perceptron neural network (MLP) classifier for our machine learning work. In the machine learning classifier module, we subscribe to the MQTT topics that the sensor’s reading was published to by the broker. Every new reading generated by the sensors serves as an instance of xtest that we use the MLP classifier for the detection of abnormality at any point in time.

However, it was noticed that the MLP classifier often misclassified largely due to overfitting and insufficient training data. We only had five sensors and we formulate our training data using different possible combination of sensor data reading scenario. Thus, we changed to decision tree which is a supervised learning method used for classification and regression. We used it to create a model that can classify any real-time sensor input data by learning simple decision rules inferred from our training data. Schematically, the decision tree can be illustrated showing all the decision rule that was implemented in the model. As shown in figure 8, we use a subset of the features to evaluate the safety condition in a smart home environment. During the training phase, we trained the model using a dataset containing different possible values for the sensors. For every MQTT message that was published, we classify it in real-time to detect if there is a problem or if it is in good condition. As shown in figure 9, the classification outputs show the effectiveness of our approach.

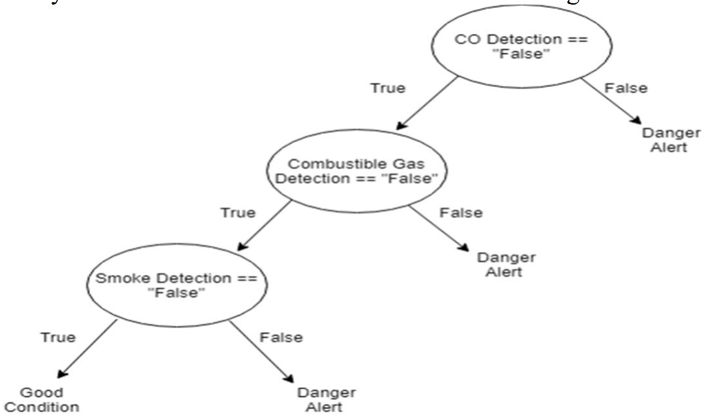
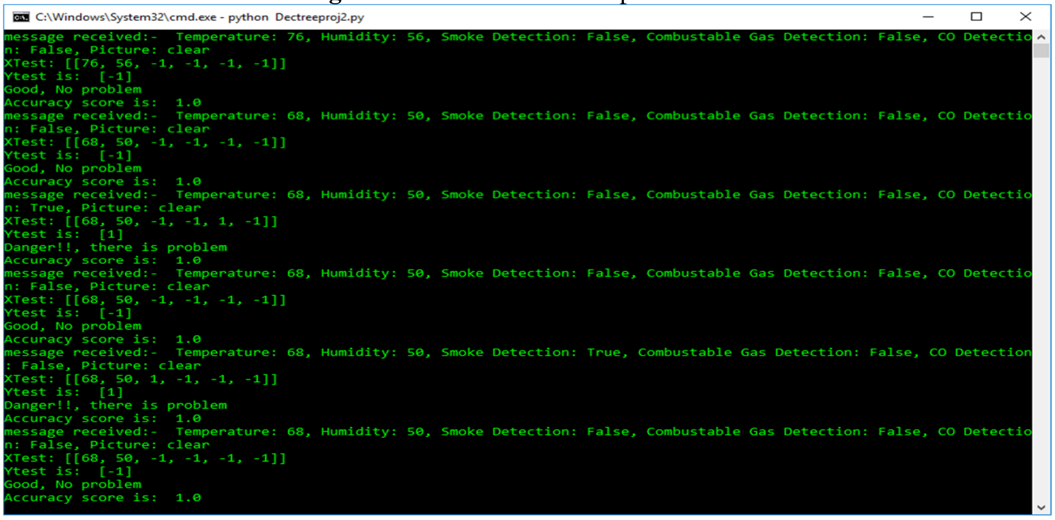


Fig 8: Decision tree model representation



```
message received:-  Temperature: 76, Humidity: 56, Smoke Detection: False, Combustable Gas Detection: False, CO Detectio
n: False, Picture: clear
XTest: [[76, 56, -1, -1, -1, -1]]
Ytest is:  [-1]
Good, No problem
Accuracy score is:  1.0
message received:-  Temperature: 68, Humidity: 50, Smoke Detection: False, Combustable Gas Detection: False, CO Detectio
n: False, Picture: clear
XTest: [[68, 50, -1, -1, -1, -1]]
Ytest is:  [-1]
Good, No problem
Accuracy score is:  1.0
message received:-  Temperature: 68, Humidity: 50, Smoke Detection: False, Combustable Gas Detection: False, CO Detectio
n: True, Picture: clear
XTest: [[68, 50, -1, -1, 1, -1]]
Ytest is:  [1]
Danger!!, there is problem
Accuracy score is:  1.0
message received:-  Temperature: 68, Humidity: 50, Smoke Detection: False, Combustable Gas Detection: False, CO Detectio
n: False, Picture: clear
XTest: [[68, 50, -1, -1, -1, -1]]
Ytest is:  [-1]
Good, No problem
Accuracy score is:  1.0
message received:-  Temperature: 68, Humidity: 50, Smoke Detection: True, Combustable Gas Detection: False, CO Detection
: False, Picture: clear
XTest: [[68, 50, 1, -1, -1, -1]]
Ytest is:  [1]
Danger!!, there is problem
Accuracy score is:  1.0
message received:-  Temperature: 68, Humidity: 50, Smoke Detection: False, Combustable Gas Detection: False, CO Detectio
n: False, Picture: clear
XTest: [[68, 50, -1, -1, -1, -1]]
Ytest is:  [-1]
Good, No problem
Accuracy score is:  1.0
```

Fig 9: Decision tree output results

6. Conclusion

Security is a major concern in IoT that needs to be addressed by all IoT device manufacturers and users. This study explored and investigated flaws inherent in an IoT system that was developed using Raspberry Pi, sensors and MQTT protocol for communication. Vulnerabilities were found in the system and active as well as passive attacks were executed to comprise the security of the IoT system. After discovering multiple vulnerabilities within the system and exploiting these vulnerabilities, protection measures were implemented to ensure data privacy and security of the system. Additionally, a real-time anomaly detection in streaming environmental sensor data was also developed to alert user of dangerous conditions so that appropriate measures can be taken for safety of lives and properties.

References

N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, thirdquarter 2019, pp. 2702-2733.

A. Hameed and A. Alomary, "Security Issues in IoT: A Survey," *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, Sakhier, Bahrain, 2019, pp. 1-5.

F. Meneghello, M. Calore, D. Zucchetto, M. Polese and A. Zanella, "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices," in *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182-8201, Oct. 2019.

R. Gurunath, M. Agarwal, A. Nandi and D. Samanta, "An Overview: Security Issue in IoT Network," *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on*, Palladam, India, 2018, pp. 104-107.

J. Bugeja, B. Vogel, A. Jacobsson and R. Varshney, "IoTSM: An End-to-end Security Model for IoT Ecosystems," *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kyoto, Japan, 2019, pp. 267-272.

J. M. Blythe and S. D. Johnson, "The consumer security index for IoT: A protocol for developing an index to improve consumer decision making and to incentivize greater security provision in IoT devices," *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, London, 2018, pp. 1-7.

O. Kayode and A. S. Tosun, "Analysis of IoT Traffic using HTTP Proxy," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1-7.

S. N. M. García, J. L. Hernández-Ramos and A. F. Skarmeta, "Test-based risk assessment and security certification proposal for the Internet of Things," *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, 2018, pp. 641-646.

H. Garg and M. Dave, "Securing IoT Devices and Securely Connecting the Dots Using REST API and Middleware," *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Ghaziabad, India, 2019, pp. 1-6.

J. Choi, Y. Shin and S. Cho, "Study on information security sharing system among the industrial IoT service and product provider," *2018 International Conference on Information Networking (ICOIN)*, Chiang Mai, 2018, pp. 551-555.

S. Naik and V. Maral, "Cyber security — IoT," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2017, pp. 764-767.